White Paper[1] on "Third Order Actuator"

R. E. McFarland

NASA, Ames Research Center

July 30, 1997

Reference is made herein to a document called "High Speed Research Reference H Cycle 3, Dynamic Aeroservoelastic (DASE) Model", by Gustav A. Taylor, May 2, 1997. In this reference document a third order actuator problem with specific limiters is posed, and a comparison is made using a couple of integration techniques, one of which, the "local linearization algorithm," comes close to the mark in terms of solving the problem. This occurs because a partial state space formulation is postulated, which I maintain is the proper approach. However, the zero order data hold that is assumed during linear regions is not an optimal choice, and the reduction of both velocities and positions to simple zero order hold integrators when limits are encountered is also non-optimal, especially because of discontinuities in the transition region. Also, the actual implementation (see p. 8-6) is different from the desired model (see p. 8-1). Finally, the performance is in error. Specifically, the displayed acceleration in Fig. 8-4(d) cannot possibly produce the velocity in Fig. 8-4(c).

I am in the process of developing a paper on "Discrete Solutions to Transfer Functions with Nonlinear State Constraints." The third order actuator problem of the reference addresses a subset of the material that I am developing, which treats arbitrary nonlinearities, including limiters, dead bands, hysteresis, *etc.*, for transfer functions of arbitrary order. As applied to the third order actuator model, my solution is presented.

The Problem

A secondary model is developed on p. 8-6 of the reference. This problem is less exacting than the originally posed problem in the reference, and almost trivially handled by the techniques of this paper (*e.g.*, see steps (7) through (12) of "The Algorithm"). Using the original model of p. 8-1 of the reference, which is more complex and presumably the desired model, a re-write of this model is presented in Fig. 1:

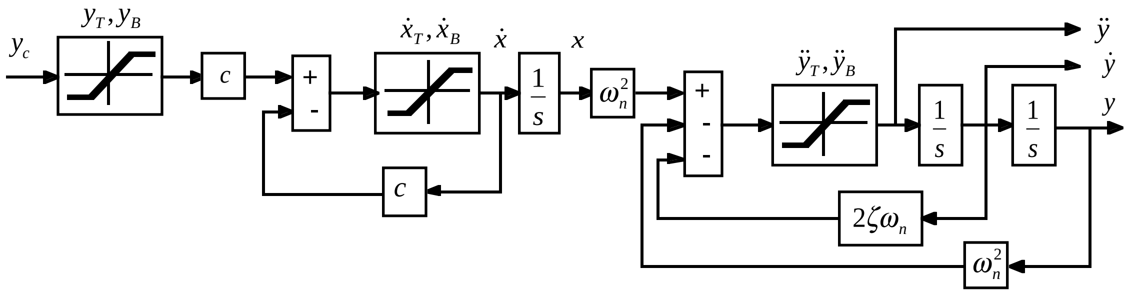

Fig. 1 - Desired Third Order Actuator Model

[1] This paper disloses new simulation technology, copyrighted by NASA. Material utilizing this technology published after this date should make reference to the contribution of Ames Research Center.

This limited system is solved using “The Algorithm,” as defined herein.

The algorithmic approach that I have developed has an underlying philosophy. This is to maintain discrete state space solutions to linear transfer functions in the presence of nonlinearities, by introducing a “pseudo command” concept, where discrete variables are created that track hypothetical commands, as required to maintain state space solutions through nonlinear regions. Because a pseudo command is then applicable in both linear and nonlinear regions, the entrance to, and exit from these regions is handled seamlessly. Also, using the triangular hold methods of this paper, all states (and their derivatives) are concurrent. Most importantly, the triangular hold methods are absolutely required in the development of z-transforms for systems where the numerator and denominator orders are equal. As will be seen, The Algorithm requires two of these particular systems for the third order actuator.

One pseudo command is usually required for each limiter (or other nonlinearity). For this problem, however, we only need two pseudo commands, because the initial limiter has a trivial solution. The algorithm presented below is for two sampled-data systems, both of which use the triangular data hold. This implementation for the problem seems to be the easiest to follow. However, the pseudo command technique may also be applied to this problem using a single sampled-data system, in an application to a combined third order Laplace system (*al la* p. 8-6 of the reference). Furthermore, an advancing data hold may be used in either the single or double sampled-data implementations (at one point only!), although this is not recommended, because the highest state derivative then lags the states by one cycle. Also, at Ames Research Center we generally develop force and moment outputs as applicable at the beginning of each cyclic interval, concurrent with pilot inputs. This usually precludes advancing integration forms from appearing in an actuator model, because an actuator invariably leads to forces and moments. Aircraft states are advanced in our generalized service routine, STRIKE.

With the problem formulation of separated first and second order systems, the acceleration limit does not influence the velocity $\dot{x}$, as shown in Fig. 1. In a total-system third order implementation (*i.e.,* the reduced system of p. 8-6 of the reference) this velocity would be influenced by feedback states from the second order transfer function.

In linear regions the originally-postulated two sampled-data system of p. 8-1 of the reference becomes as shown in Fig. 2,

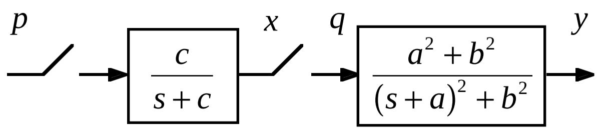

Fig. 2 - The Linear Model

where the coefficients are related to the system coefficients by,

$$c = 1/\tau$$

$$a = \zeta\omega_n$$

$$b = \omega_n\sqrt{1-\zeta^2}$$

$$\omega_n^2 = a^2 + b^2$$

In preparation for "The Algorithm," consider the transformations of Table I, which include the derivatives of the first and second order systems in Fig. 2 up to the order of the denominators.

Laplace Notation	z-Transform Notation
$\frac{x(s)}{p(s)} = \frac{c}{s+c}$	$\frac{x(z)}{p(z)} = \frac{e_1 + e_2 z^{-1}}{1 - d_1 z^{-1}}$
$\frac{\dot{x}(s)}{p(s)} = \frac{cs}{s+c}$	$\frac{\dot{x}(z)}{p(z)} = \frac{e_3 + e_4 z^{-1}}{1 - d_1 z^{-1}}$
$\frac{y(s)}{q(s)} = \frac{a^2+b^2}{(s+a)^2+b^2}$	$\frac{y(z)}{q(z)} = \frac{h_1 + h_2 z^{-1} + h_3 z^{-2}}{1 - g_1 z^{-1} - g_2 z^{-2}}$
$\frac{\dot{y}(s)}{q(s)} = \frac{(a^2+b^2)s}{(s+a)^2+b^2}$	$\frac{\dot{y}(z)}{q(z)} = \frac{h_4 + h_5 z^{-1} + h_6 z^{-2}}{1 - g_1 z^{-1} - g_2 z^{-2}}$
$\frac{\ddot{y}(s)}{q(s)} = \frac{(a^2+b^2)s^2}{(s+a)^2+b^2}$	$\frac{\ddot{y}(z)}{q(z)} = \frac{h_7 + h_8 z^{-1} + h_9 z^{-2}}{1 - g_1 z^{-1} - g_2 z^{-2}}$

Table I - Triangular Data Hold z-Transforms

The z-transform coefficients shown in Table I are given in Appendix A.

The Algorithm

(1) Perform the trivial initial limit on the input y_c. This produces the initial value (per cycle) of the first pseudo command p_k.

$$p_k = \begin{cases} y_T & \text{if } y_c \geq y_T \\ y_B & \text{if } y_c \leq y_B \\ y_c & \text{otherwise} \end{cases}$$

(2) Using the pertinent triangular data hold coefficients, solve for the derivative of the first order system, where its value may encounter the nonlinearity.

$$\dot{x}_k = d_1 \dot{x}_{k-1} + e_3 p_k + e_4 p_{k-1}$$

This derivative is concurrent with the input y_c.

(3) Determine if $\dot{x}_k$ is on or beyond a limit. If so, replace it.

$$\dot{x}_k = \begin{cases} \dot{x}_T & \text{if } \dot{x}_k \geq \dot{x}_T \\ \dot{x}_B & \text{if } \dot{x}_k \leq \dot{x}_B \\ \dot{x}_k & \text{otherwise} \end{cases}$$

(4) If no replacement has occurred, this step can be ignored, because the first order system is in a linear region. Otherwise, replace the first pseudo command p_k to conform to the difference equation in step (2), using the replaced $\dot{x}_k$.

$$p_k = \left(\dot{x}_k - d_1\dot{x}_{k-1} - e_2 p_{k-1}\right)/e_3$$

This modifies the first pseudo command, which differs from the original command (per cycle) whenever a nonlinear region is encountered. With this pseudo command the state space solution in step (2) is exact, and the system remains linear.

(5) Since the derivative is linear (using the pseudo command), its integral is linear (also using the pseudo command). Using the appropriate triangular hold coefficients to produce a concurrent output[2], compute the state space solution for the output of the first order system, using the first pseudo command[3], as possibly modified.

$$x_k = d_1 x_{k-1} + e_1 p_k + e_2 p_{k-1}$$

(6) At this point the first order system is completed. Either here, or at the end of the algorithm (step 12), we must ladder down the cyclic values in preparation for the next cycle.

$$x_{k-1} = x_k$$
$$\dot{x}_{k-1} = \dot{x}_k$$
$$p_{k-1} = p_k$$

(7) Initialize the pseudo command for the second order system to the output of the first order system.

$$q_k = x_k$$

(8) As in step (2), using the pertinent triangular data hold coefficients, solve for the acceleration of the second order system, where its value may encounter the nonlinearity.

$$\ddot{y}_k = g_1\ddot{y}_{k-1} + g_2\ddot{y}_{k-2} + h_7 q_k + h_8 q_{k-1} + h_9 q_{k-2}$$

This derivative is also concurrent with the input y_c, providing an advancing algorithm is not used in step (5).

(9) Determine if $\ddot{y}_k$ is on or beyond a limit. If so, replace it.

$$\ddot{y}_k = \begin{cases} \ddot{y}_T & \text{if } \ddot{y}_k \ge \ddot{y}_T \\ \ddot{y}_B & \text{if } \ddot{y}_k \le \ddot{y}_B \\ \ddot{y}_k & \text{otherwise} \end{cases}$$

[2] Or, at this point, first order hold coefficients to produce an advanced output.
[3] Here we have the core of The Algorithm. Integrals in discrete form are driven by the modified commands, not by the derivatives. Thus, all subsystems retain their linear solution form. Extraneous assumptions are not required in nonlinear regions.

(10) If no replacement is required, this step can also be ignored, because the second order system is in a linear region. Otherwise, replace the second pseudo command q_k to conform to the difference equation in step (8), using the replaced $\ddot{y}_k$.

$$q_k = \left(\ddot{y}_k - g_1\ddot{y}_{k-1} - g_2\ddot{y}_{k-2} - h_8 q_{k-1} - h_9 q_{k-2}\right)/h_7$$

This modifies the second pseudo command. With this pseudo command the state space solution in step (8) is exact, and the system remains linear.

(11) Since the second derivative is linear (using the second pseudo command), all integrals are linear (also using the second pseudo command). Using the appropriate triangular hold coefficients to produce concurrent outputs[4], compute the state space solutions for the velocity and position outputs of the second order system, using the second pseudo command.

$$\dot{y}_k = g_1\dot{y}_{k-1} + g_2\dot{y}_{k-2} + h_4 q_k + h_5 q_{k-1} + h_6 q_{k-2}$$

$$y_k = g_1 y_{k-1} + g_2 y_{k-2} + h_1 q_k + h_2 q_{k-1} + h_3 q_{k-2}$$

(12) At this point the second order system is also complete. Ladder down the final cyclic values in preparation for the next cycle.

$$\ddot{y}_{k-2} = \ddot{y}_{k-1}$$
$$\ddot{y}_{k-1} = \ddot{y}_k$$
$$\dot{y}_{k-2} = \dot{y}_{k-1}$$
$$\dot{y}_{k-1} = \dot{y}_k$$
$$y_{k-2} = y_{k-1}$$
$$y_{k-1} = y_k$$
$$q_{k-2} = q_{k-1}$$
$$q_{k-1} = q_k$$

Note that the algorithm produces all states and their derivative, and all values are concurrent with the input y_c at the time $t = kh$. This is very important in the distribution of these variables to other simulation components.

Subroutine TOAD, in FORTRAN

Many efficiencies occur in programming The Algorithm for the third order system of Fig. 1. Also, initialization can be a problem, especially if the initial command induces limited states. For these reasons Subroutine TOAD (Third Order Actuator Design) is provided in Appendix B. Note that the system coefficients may be nonstationary, providing that they conform to the "slowly varying coefficient hypothesis," were it is assumed that the digital technology exists to implement transition intervals which are small enough so that

[4] See footnote 2. If an advance (*e.g.,* a first order data hold) is used on the first order system, then it must not be used here!

coefficient derivatives may be ignored in the treatment of the discrete solution as a recursive boundary problem.

In Subroutine TOAD the quantity Mode controls the mode of operations. If Mode is negative, the program is in initialization mode where it is assumed that $t = 0$. A zero value for Mode denotes Hold mode, where no operations are performed. A value of unity for Mode is the conventional Operate mode, where time (and the solution) advances. A value for Mode that is greater than unity also denotes Operate Mode (actually, Operate Mode Plus), with the overhead of re-computing the z-transform coefficients because, presumably, some system coefficient has changed sufficiently to warrant this re-evaluation. This Mode may be exercised "periodically" during conventional Operate mode operations for the case of nonstationary coefficients.

Appendix C consists of a sample batch driver program to TOAD, where its outputs consist of an IGOR text file. This program, called DTOAD, has two options. The first option is to replicate (and improve) the data provided by the reference document. The outputs of this process are given in Figures 3 and 4. The second option introduces a different excitation profile, which more exhaustively exercises the discontinuities, and also uses different parameters, because the original parameters are much too fast for a visualization of the effects of entering and leaving discontinuities. The outputs of this system are given in Figs. 5 and 6. The pertinent data is shown in the driver routine of Appendix C.

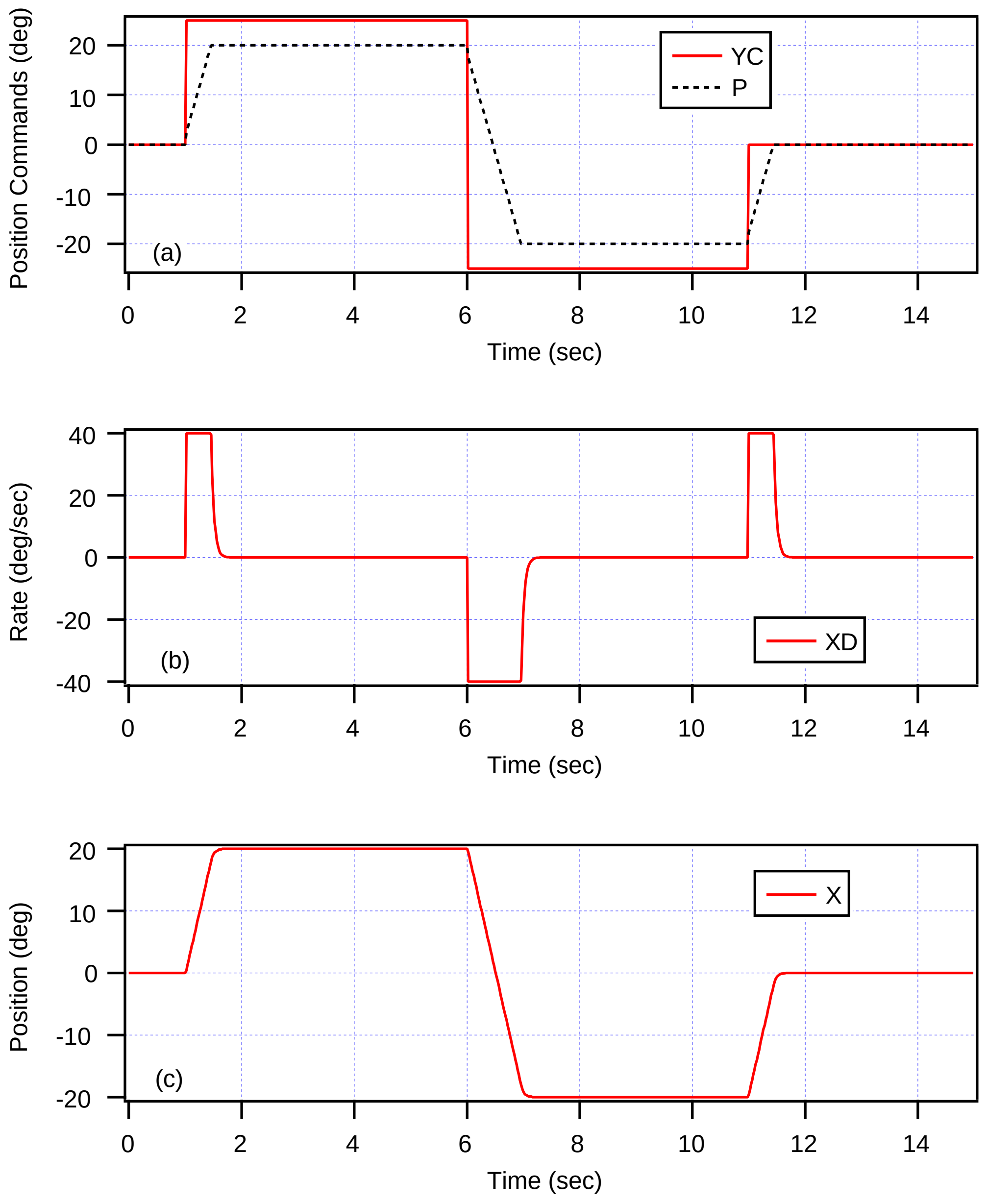

Fig. 3 - Original Parameters, First Order System Outputs

In Fig. 3(a) the input command is beyond the input limiter values, and its rate of change is for the most part beyond the rate limiter. Note how the first pseudo command tracks a behavior compatible with both of these limiters.

In Fig. 3(b) the limited rate command is available. In Fig. 3(c) we see that the resultant output of the first order system tracks the original command within the limitations of its functionality.

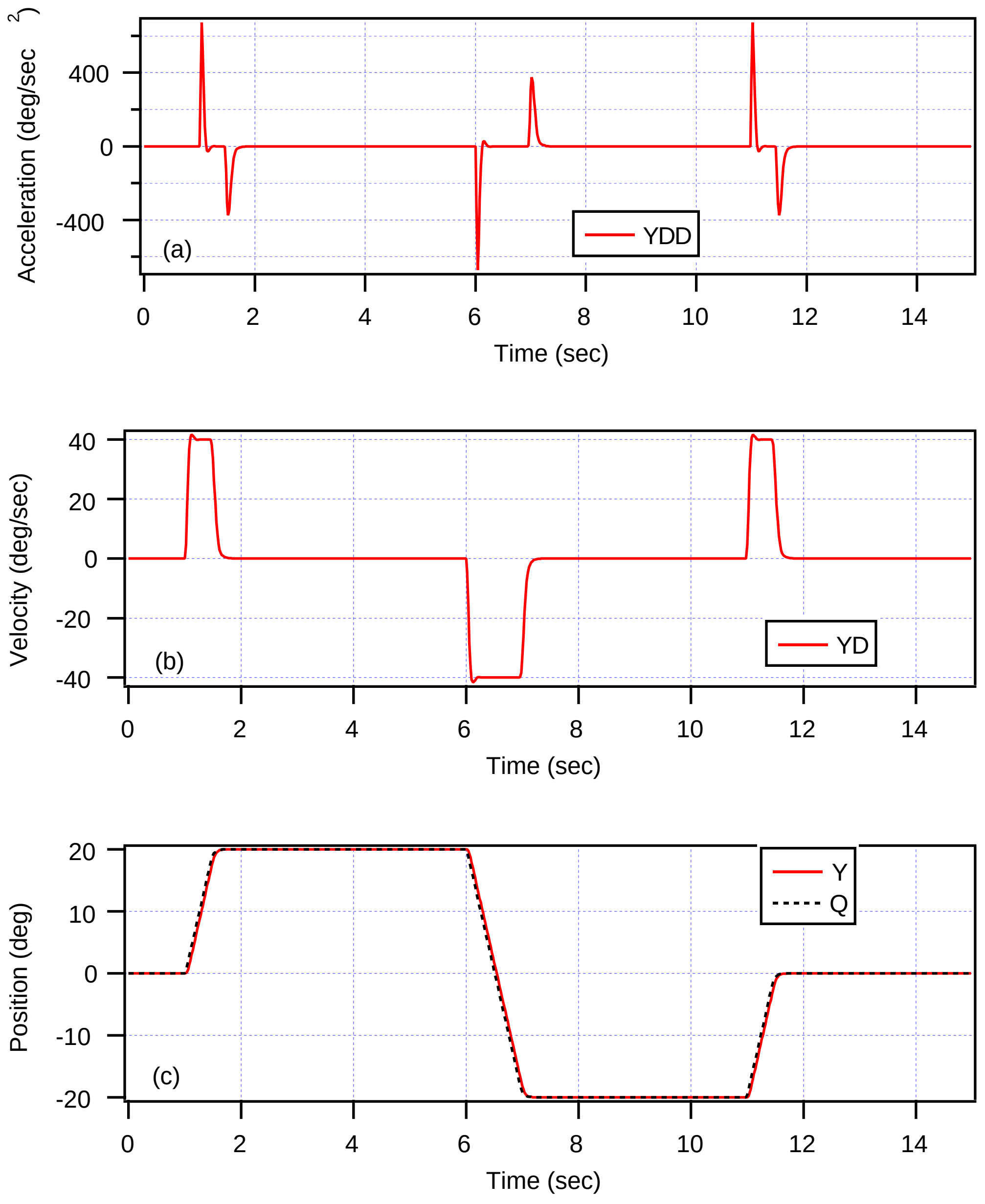

Fig. 4 - Original Parameters, Total System Outputs

In Fig. 4(a) the acceleration of the second order system is available. It does not have magnitudes anywhere near those given in the reference document. Indeed, these accelerations produce the velocity output of Fig. 4(b), and the final position output of Fig. 4(c).

Except to display that accelerations are accurately produced, the input drive profile and parameters of this system do not reveal much of the true capacity of TOAD. For that reason Figs. 5 and 6 were created, showing a much more active input profile, a larger time constant, and a slower second order system.

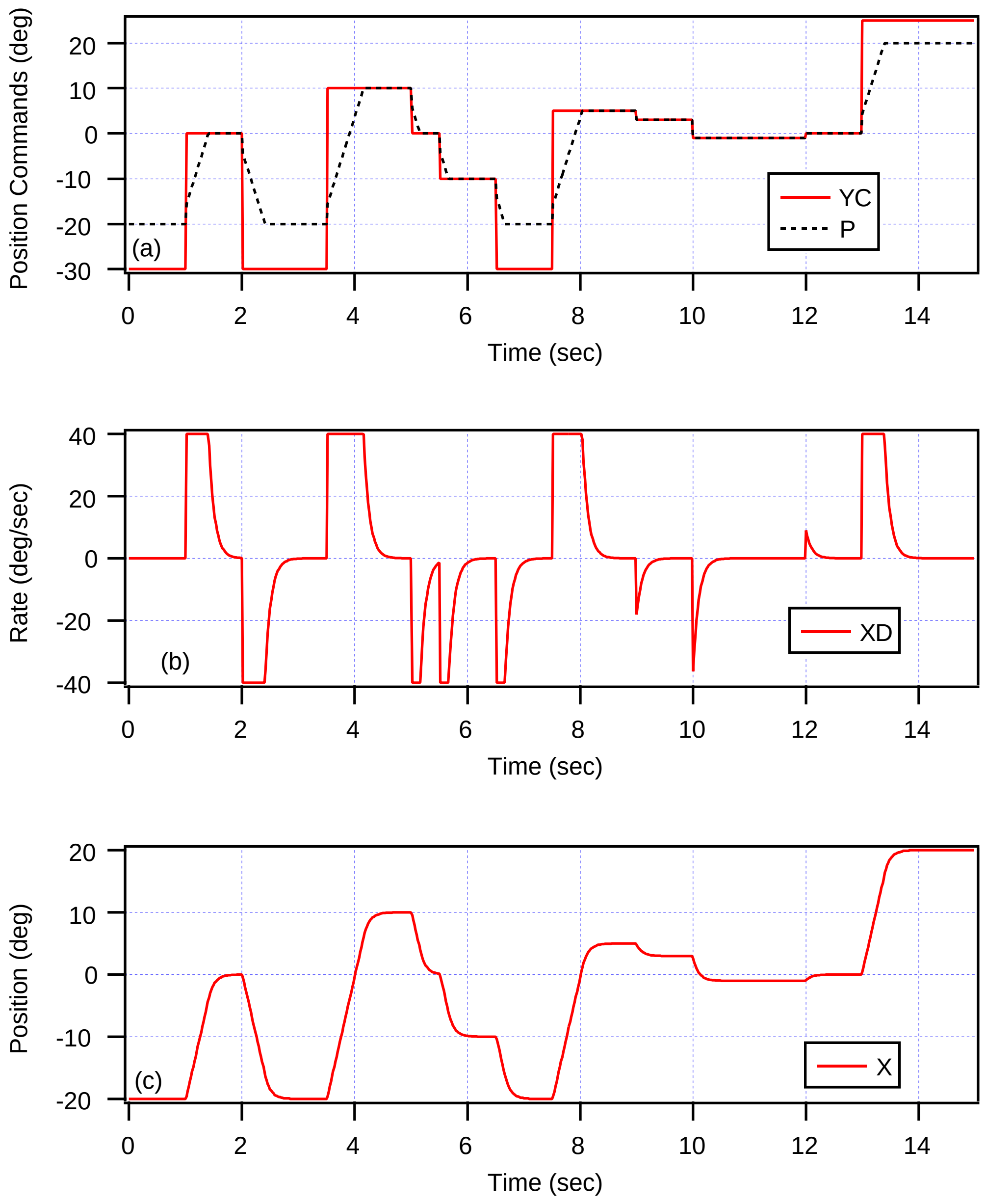

Fig. 5 - Modified Parameters, First Order System Outputs

In Fig. 5(a) both the linear and nonlinear regions of the input are exercised. They produce an interesting track for the pseudo command. Note that the initial value is also limited.

The pseudo command variations in Fig. 5(a) produce the velocities for the first order system as shown in Fig. 5(b). Here both linear and nonlinear regions are included.

The resultant output of the first order system is given in Fig. 5(c). Clearly it attempts to follow the input command, as curtailed by both the position and velocity limits.

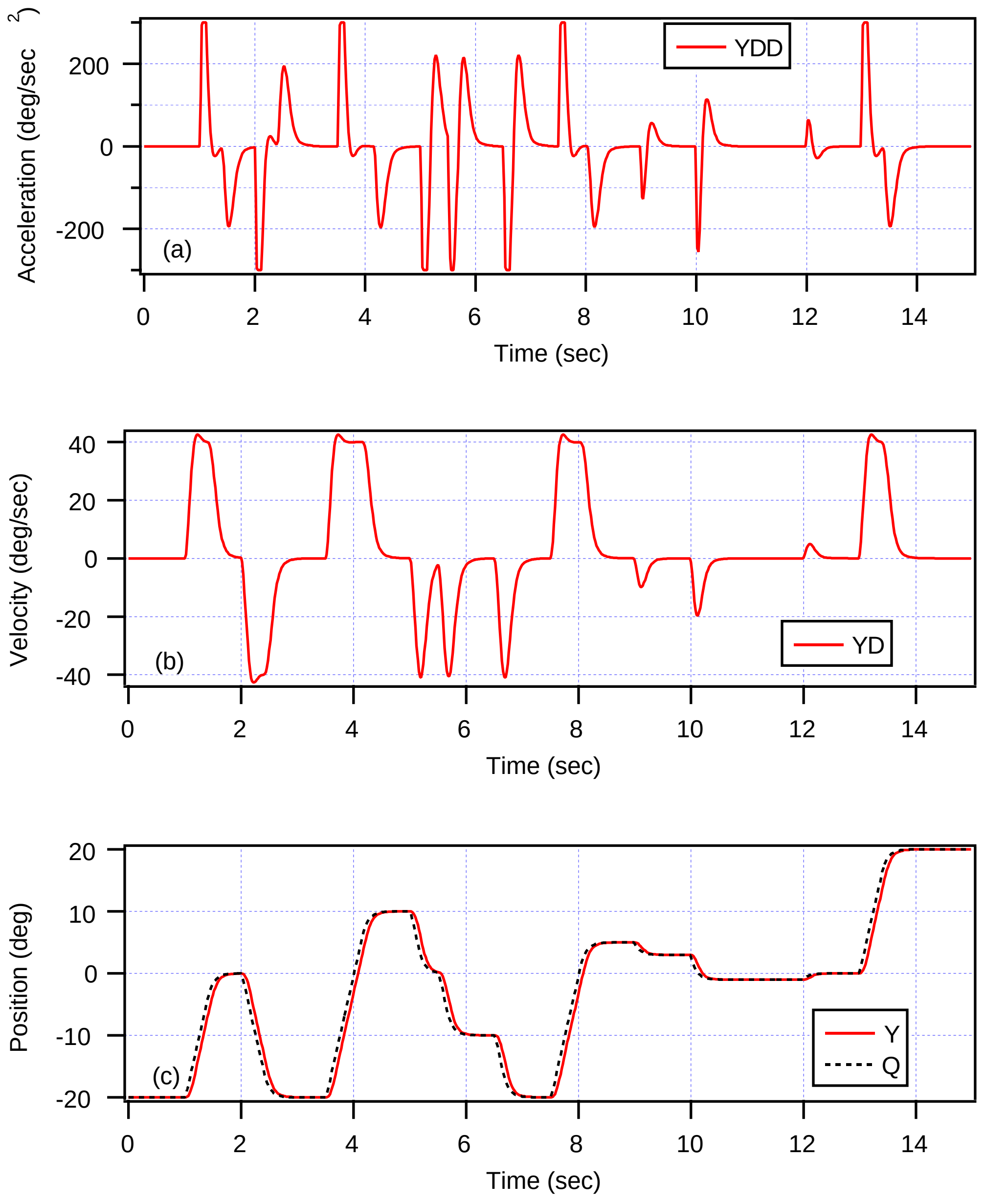

Fig. 6 - Modified Parameters, Total System Outputs

As driven by the command profile of Fig. 5(c), we see that the acceleration of the second order system, as shown in Fig. 6(a), is quite dynamic. Note that the limiter has been moved down from 10000 (!) in the original problem, to a value of 300.

The velocity history of Fig. 6(b) is also quite dynamic. It would be interesting to add another limiter here... but that's not in the original problem.

Compare the final output of Fig. 6(c) with the actuator command of Fig. 5(a). It is maintained that the subroutine TOAD represents a class solution to the third order actuator problem.

Timing of TOAD

In order to time the TOAD subroutine we compare its operate cycle time to that of a simple sine function operation. For 500 different runs, these operations were performed 10,000 times each. There is a good deal of variability on the computer that was used, because of other users, *etc.*, but the data is fairly consistent, as shown in Fig. 7. The entire third order actuator problem may be solved in an equivalent computer overhead of three calls to the system's sine function.

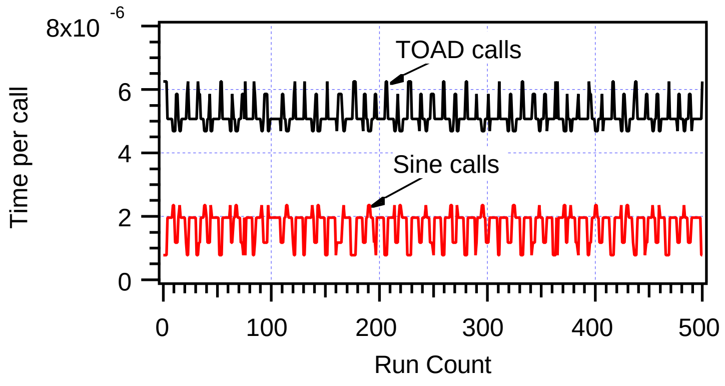

Fig. 7 - Timing the TOAD Subroutine

Conclusions

A program has been developed for a very efficient and very accurate discrete solution to the third order actuator problem addressed in the reference. This program utilizes new technology developed by NASA, and is submitted for review by the simulation community.

The techniques disclosed here have wide ramifications for the discrete simulation of a much more generalized class of Laplace systems containing various types of discontinuities.

Appendix A

z-Transform Coefficients

These coefficients utilize the triangular data hold assumption. The resultant difference equation for each form is also presented.

$$d_1 = e^{-hc}$$

$$e_1 = \frac{1}{hc}\left(e^{-hc} + hc - 1\right)$$

$$e_2 = \frac{1}{hc}\left[1 - e^{-hc}(1 + hc)\right]$$

$$x_k = d_1 x_{k-1} + e_1 p_k + e_2 p_{k-1}$$

$$e_3 = \frac{1}{h}\left(1 - e^{-hc}\right)$$

$$e_4 = -\frac{1}{h}\left(1 - e^{-hc}\right)$$

$$\dot{x}_k = d_1 \dot{x}_{k-1} + e_3 p_k + e_4 p_{k-1}$$

$$g_1 = 2e^{-ah}\cos bh$$

$$g_2 = -e^{-2ah}$$

$$h_1 = 1 - \frac{2a}{h\omega_n^2}\left(1 - e^{-ah}\cos bh\right) + \left(\frac{a^2 - b^2}{bh\omega_n^2}\right)e^{-ah}\sin bh$$

$$h_2 = \frac{2a}{h\omega_n^2}\left(1 - e^{-2ah}\right) - 2e^{-ah}\cos bh - 2\left(\frac{a^2 - b^2}{bh\omega_n^2}\right)e^{-ah}\sin bh$$

$$h_3 = e^{-2ah}\left(1 + \frac{2a}{h\omega_n^2}\right) - \frac{2a}{h\omega_n^2}e^{-ah}\cos bh + \left(\frac{a^2 - b^2}{bh\omega_n^2}\right)$$

$$y_k = g_1 y_{k-1} + g_2 y_{k-2} + h_1 q_k + h_2 q_{k-1} + h_3 q_{k-2}$$

$$h_4 = \frac{1}{h}\left(1 - e^{-ah}\cos bh - \frac{a}{b}e^{-ah}\sin bh\right)$$

$$h_5 = \frac{1}{h}\left(e^{-2ah} + \frac{2a}{b}e^{-ah}\sin bh - 1\right)$$

$$h_6 = \frac{1}{h}\left(e^{-ah}\cos bh - \frac{a}{b}e^{-ah}\sin bh - e^{-2ah}\right)$$

$$\dot{y}_k = g_1\dot{y}_{k-1} + g_2\dot{y}_{k-2} + h_4 q_k + h_5 q_{k-1} + h_6 q_{k-2}$$

$$h_7 = \frac{\omega_n^2 e^{-ah}\sin bh}{bh}$$

$$h_8 = -2\frac{\omega_n^2 e^{-ah}\sin bh}{bh}$$

$$h_9 = \frac{\omega_n^2 e^{-ah}\sin bh}{bh}$$

$$\ddot{y}_k = g_1\ddot{y}_{k-1} + g_2\ddot{y}_{k-2} + h_7 q_k + h_8 q_{k-1} + h_9 q_{k-2}$$

Appendix B

Subroutine TOAD

```
C TOAD.FOR
C Third Order Actuator Design

C R. E. McFarland, NASA, Ames Research Center
C July 29, 1997

      SUBROUTINE TOAD(MODE,H,TAU,WN,ZETA,BV,TV,YC,X,XD,Y,YD,YDD,P,Q)

      DIMENSION BV(3),TV(3)

      IF(MODE) 10,120,20

  10    PP = YC
        IF(PP.GT.TV(1)) PP = TV(1)
        IF(PP.LT.BV(1)) PP = BV(1)

        XDP = 0.0
        X = PP

  20  IF(MODE.EQ.1) GO TO 30

        AHI = TAU/H
        D1 = EXP(-1.0/AHI)
        E1 = 1.0 - (1.0 - D1)*AHI
        E2 = AHI - (1.0+AHI)*D1
        E3 = (1.0 - D1)/H
        E4 = - E3

  30    P = YC

        IF(P.GT.TV(1)) P = TV(1)
        IF(P.LT.BV(1)) P = BV(1)

        TEM = D1*XDP + E4*PP
        XD = TEM + E3*P

        IF(XD.LT.TV(2)) GO TO 40
        XD = TV(2)
        GO TO 50

  40    IF(XD.GT.BV(2)) GO TO 60
        XD = BV(2)
  50    P = (XD - TEM)/E3

  60  X = D1*X + E1*P + E2*PP

      XDP = XD
      PP = P

      Q = X
      IF(MODE) 70,120,80
 70     QQ = Q
        QQQ = Q
        YDD = 0.0
        YDDP = 0.0
        YDDPP = 0.0
        YD = 0.0
        YDP = 0.0
        YDPP = 0.0
        Y = Q
        YP = Q
        YPP = Q

 80   IF(MODE.EQ.1) GO TO 90
        A = ZETA*WN
        B = WN*SQRT(1.0-ZETA**2)
        W2 = WN**2
        AH = A*H
        BH = B*H
        EX = EXP(-AH)
        EXCBH = EX*COS(BH)
        G1 = 2.0*EXCBH
        G2 = - EX**2
        EXSBH = EX*SIN(BH)
        H7 = W2*EXSBH/BH
        H8 = - 2.0*H7
        H9 = H7
        AOB = A/B
        H4 = (1.0 - EXCBH - AOB*EXSBH)/H
        H5 = (- G2 - 1.0 +2.0*AOB*EXSBH)/H
        H6 = (EXCBH + G2 - AOB*EXSBH)/H
        HW2 = 1.0/(H*W2)
        TAOV = 2.0*A*HW2
        DOV = (A**2 - B**2)*EXSBH*HW2/B
        H1 = 1.0 - TAOV*(1.0 - EXCBH) + DOV
        H2 = TAOV*(1.0 + G2) - 2.0*EXCBH - 2.0*DOV
        H3 = - G2 - (G2 + EXCBH)*TAOV + DOV

 90     CONTINUE

        TEM = G1*YDDP+G2*YDDPP+H8*QQ+H9*QQQ
        YDD = TEM+H7*Q

        IF(YDD.GE.TV(3)) THEN
          YDD = TV(3)
          GO TO 100
        END IF

        IF(YDD.GT.BV(3)) GO TO 110
        YDD = BV(3)
100     Q = (YDD - TEM)/H7

110     YD = G1*YDP + G2*YDPP + H4*Q + H5*QQ + H6*QQQ
```

```
       Y = G1*YP + G2*YPP + H1*Q + H2*QQ +H3*QQQ

       YPP = YP
       YP = Y
       YDPP = YDP
       YDP = YD
       YDDPP = YDDP
       YDDP = YDD
       QQQ = QQ
       QQ = Q
120  RETURN
     END
```

Appendix C

Driver Routine DTOAD

```
C DTOAD.FOR DRIVES TOAD.FOR
C OUTPUT:  PREPARES AN IGOR TEXT FILE.
C
C R. E. MCFARLAND, NASA, AMES RESEARCH CENTER
C JULY 29, 1997

C LINK DTOAD,TOAD
C RUN DTOAD

      DIMENSION TV(3),BV(3)

      TYPE 80
  80  FORMAT(' 1 = ORIGINAL DATA?')
      READ 85,IANS
  85  FORMAT(I1)

      IF(IANS.EQ.1) THEN
        TAU = 0.05
        H = 0.02
        BV(1) = - 20.0
        BV(2) = - 40.0
        BV(3) = - 10000.0
        TV(1) = 20.0
        TV(2) = 40.0
        TV(3) = 10000.0
        WN = 40.0
        ZETA = 0.7071
      ELSE
        TAU = 0.1
        H = 0.02
        BV(1) = -20.0
        BV(2) = -40.0
        BV(3) = -300.0
        TV(1) = 20.0
        TV(2) = 40.0
        TV(3) = 300.0
        WN = 20.0
        ZETA = 0.7071
      END IF

      OPEN(25,FILE='TOAD.DAT',STATUS='UNKNOWN')

      WRITE(25,120)
 120  FORMAT('IGOR'/'WAVES TT,PC,U,UD,V,VD,VDD,PK,QK'/'BEGIN')

      TIME = 0.0
      MODE = -1
      TSTOP = 15.0
      ISTOP = TSTOP/H
      DO I=1,ISTOP
        IF(IANS.NE.1) THEN
          YC = -30.0
          IF(TIME.GE.1.0) YC = 0.0
          IF(TIME.GE.2.0) YC = -30.0
          IF(TIME.GE.3.5) YC = 10.
          IF(TIME.GE.5.0) YC = 0.0
          IF(TIME.GE.5.5) YC = -10.
          IF(TIME.GE.6.5) YC = -30.0
          IF(TIME.GE.7.5) YC = 5.0
          IF(TIME.GE.9.0) YC = 3.0
          IF(TIME.GE.10.0) YC = -1.0
          IF(TIME.GE.12.0) YC = 0.0
          IF(TIME.GE.13.0) YC = 25.0
        ELSE
          YC = 0.0
          IF(TIME.GE.1.0) YC = 25.0
          IF(TIME.GE.6.0) YC = -25.0
          IF(TIME.GE.11.0) YC = 0.0
        END IF

        CALL TOAD(MODE,H,TAU,WN,ZETA,BV,TV,YC,X,XD,Y,YD,YDD,P,Q)

        WRITE(25,140) TIME,YC,X,XD,Y,YD,YDD,P,Q
        TIME = TIME+H
        MODE = 1
      END DO

 140  FORMAT(F8.3/8E10.3)
      WRITE(25,150)
 150  FORMAT('END')
      CLOSE(25)

      STOP
      END
```